Introduction to STAN

A probabilistic programming language

Songpeng Zu

12 October 2020

What is STAN [1]

- A *programming language*
  - the syntax much like C++
  - it's written in C++
  - it's needed to compile and run
- Specifically designed for the *statistical modeling*
  - Vector, matrix, array and their operations
  - A series of probabilistic functions
  - A series of blocks to describe a statistical model.
  - Support sampling, maximum likelihood estimation and variational inference.

[1] https://mc-stan.org

How STAN works?

- *Hamilton Monte Carlo (HMC)* provides a general sampling procedure for Bayesian inference
  Gradient of the log probabilistic density function over the random variables are needed.
- STAN owns a mathematical library [See Carpenter et al., 2015] that can automatically get the gradients.
- Furthermore, any bounded parameters will be transformed into a non-bounded parameter space.

How to use STAN?

- Firstly, we need to write the STAN script ended with .stan to describe the data we have, the parameters, and the joint distribution of the parameters and the data.
- Secondly, we need to compile and run the codes, and analyze the results.

Show me an example

Let's consider a simple example.

- Suppose we have N binary observations $y_1, y_2, ..., y_N$. They are the *i.i.d* samples from a $Bernoulli$ distribution under the parameter θ.
- Our goal is to infer θ.

Set up the STAN env in R.

```
library(cmdstanr)
# Note: the cmdstan home path is from my computer.
set_cmdstan_path(path = paste(Sys.getenv("HOME"),
                  "softwares",
                  "cmdstan-2.23.0", sep = "/"))
library(bayesplot)
library(posterior)

cmdstan_path()
[1] "/Users/beyondpie/softwares/cmdstan-2.23.0"
cmdstan_version()
[1] "2.23.0"
```

STAN script

```
bern_mod <- cmdstan_model("bernoulli.stan",
                          ## STAN need to be complied.
                          compile = TRUE)
## show the content in the stan script.
bern_mod$print()
data {
  int<lower=0> N;
  int<lower=0,upper=1> y[N];
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
    // uniform prior on interval 0, 1
  theta ~ beta(1,1);
  y ~ bernoulli(theta);
}
```

Let's feed it some data I

```
bern_data  <- list(N = 10, y = c(0,1,0,0,0,0,0,0,0,1))
## Run MCMC using the 'sample' method
bern_mcmc <- bern_mod$sample(data = bern_data,
                             seed = 355113,
                             chains = 4,
                             parallel_chains = 2,
                             show_message = FALSE)
```

Summary of the sampling in STAN

```
## use `posterior` package
bern_mcmc$summary("theta", "mean", "sd" , "rhat",
                  "ess_bulk")
```

variable	mean	sd	rhat	ess_bulk
theta	0.2523255	0.1232632	1.001693	1425.133

Posterior draws I

```
draws <- bern_mcmc$draws()
str(draws)
 'draws_array' num [1:1000, 1:4, 1:2] -6.75 -6.97 -7.04 -7.15 -6
 - attr(*, "dimnames")=List of 3
  ..$ iteration: chr [1:1000] "1" "2" "3" "4" ...
  ..$ chain    : chr [1:4] "1" "2" "3" "4"
  ..$ variable : chr [1:2] "lp__" "theta"
## use posterior::as_draws_df to transorm the results
## into data.frame.
```

How about variational inference?

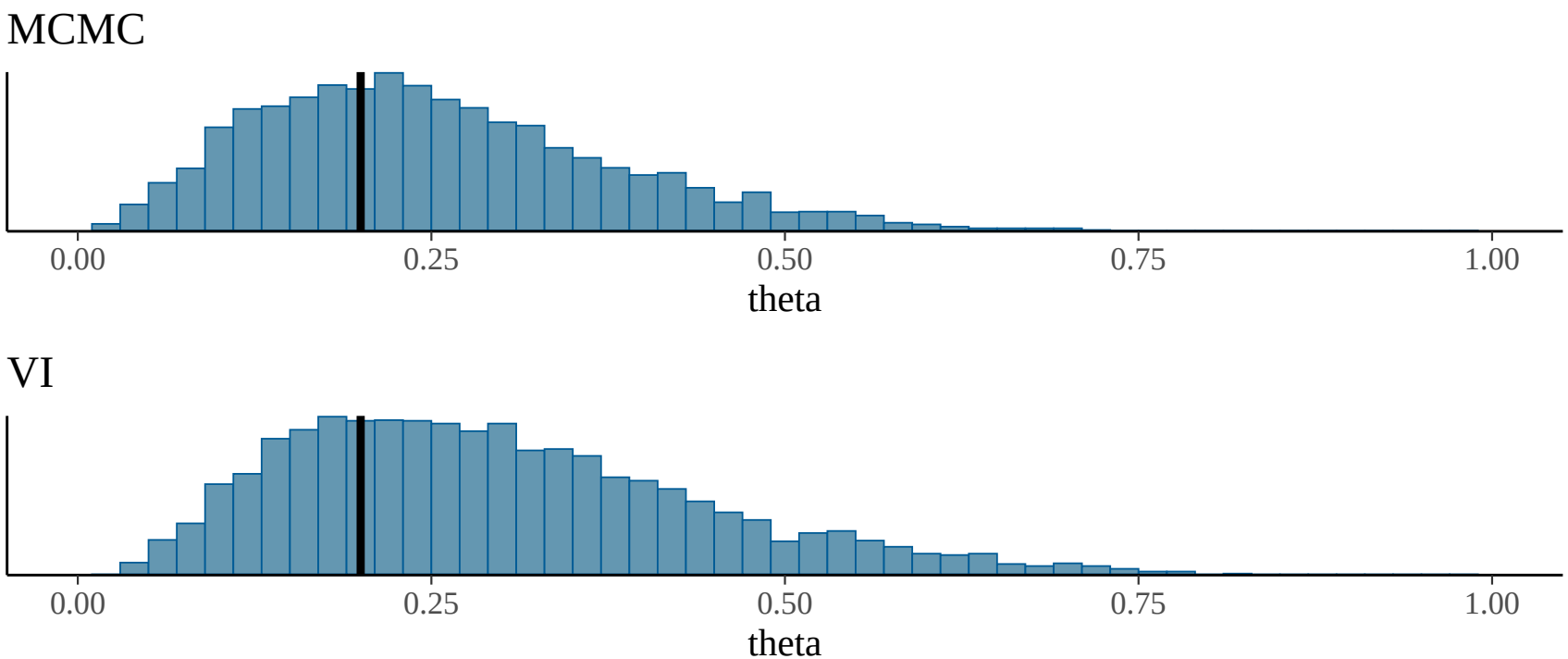
```
bern_vb <- bern_mod$variational(data = bern_data,
                                seed = 355113,
                                output_samples = 4000)
```

How about MLE estimation?

```
bern_mle <- bern_mod$optimize(data = bern_data,
                              seed = 355113)
```

Result Summary

```
bayesplot_grid(
  mcmc_hist(bern_mcmc$draws("theta"), binwidth = 0.02) +
  vline_at(bern_mle$mle(), size = 1.2),
  mcmc_hist(bern_vb$draws("theta"), binwidth = 0.02) +
  vline_at(bern_mle$mle(), size = 1.2),
  titles = c("MCMC", "VI"),
  xlim = c(0,1))
```



STAN Materials

- STAN Functions
  - Use this as the reference materials. When you want some functions, just search it.
- The STAN Language Sytanx
  - You can scan this if you want to know the whole picture of STAN syntax.
- The User Guide
  - After the introduction, you could read this document smoothly.
  - Lots of examples cover different statistical modelings.
  - It could be a good material to learn statistical models.

Thanks!

- You can find this presentation at https://github.com/beyondpie/intro_to_stan.
- Any suggesions or Pull Requests are welcome.

Bob Carpenter, Matthew D Hoffman, Marcus Brubaker, Daniel Lee, Peter Li, and Michael Betancourt. The stan math library: Reverse-mode automatic differentiation in c++. *arXiv preprint arXiv:1509.07164*, 2015.